PIXIE-4 Online Help

Version 1.41, February 2007

XIA LLC

31057 Genstar Road
Hayward, CA 94544 USA

Phone: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com

Disclaimer

- **Getting Help for the Pixie-4**

  There are several ways to get help for the Pixie-4. You can use IGOR's built-in help browser to access the Pixie-4 specific help file by selecting Help -> Help Topics from the top menu bar. Choose "Pixie4-Help" in the popup menu on the left, and select the appropriate help topic from the list on the right.

  Each Pixie-4 Run Control Panel also has a "Help" button, which directly displays the help topic for that panel. In the help topics, click on blue underlined links to jump to cross references.

- **Getting Started**

  Preparations

  1. If you have a remote controller, first install the driver software for the controller itself.
  2. Unless already installed, power down the host computer, install the controller in both the host computer and chassis, and power up the system again (chassis first).
  3. Install Wavemetrics IGOR Pro.
  4. Install the Pixie-4 software from XIA (see file readme.txt on CD-ROM).
  5. Install the Pixie-4 modules and system controller in the PXI/CompactPCI crate with power switched off. Connect detector signals to the Pixie-4 modules using the cables supplied by XIA. Then switch on power of the chassis.
  6. After the controller booted, find *Pixie4.pxp* in the installed folder and double-click it to open the Pixie-4 Viewer.

  Initial Startup

  When the Pixie Viewer has been loaded, the *Pixie Start Up Panel* should be prominently displayed in the middle of the desktop. It will prompt you to do the following:

  Select PXI chassis type
  First select the PXI chassis you are using. Currently supported are standard 4-slot or 8-slot chassis and the National Instrument 14-slot or 18-slot chassis with 3 PCI bus segments. (The 14-slot chassis has the same PCI bus configuration as the 18-slot chassis, only fewer slots)

  Specify the Pixie-4 modules:
  Next select the number of Pixie-4 modules in the system. Then specify the PXI slot number in which each module resides.
  Note: If using the 18-slot chassis, the Pixie-4 module designated as module 0 must reside in the first bus segment (any of slots 1-6).

  Finally, click [**Start Up System**]. In the IGOR history window, a message will show if the modules have been initialized successfully. You will now see the main *Pixie4 Run Control* Panel from which all work is conducted. The tabs in the *Pixie4 Run Control* Panel are arranged in logical order from left to right. For most of the actions the Pixie Viewer interacts with one Pixie-4 module at a time. The number of that module is displayed at the top right corner of the *Pixie4 Run Control* Panel (inside the [**Module**] control). Next to the [**Module**] control is the [**Channel**] control which specifies the current channel the Pixie Viewer is interacting with. The [**Module**] and the [**Channel**] are the target for all actions executed from the Pixie Viewer.

  Proceed with the steps below to configure your system.

1. In the Calibrate tab, click on the Oscilloscope button.

   This opens a graph that shows the untriggered signal input. Click [**Refresh**] to update the display. The pulses should fall between about 1600 and 15000 on the left axis. If no pulses are visible or if they are cut off above 16384 or below 0, click [**Adjust Offsets**] to automatically set the DC offset. There is a control called [**Baseline**] on the Oscilloscope which can be used to adjust the DC offset for each channel. If the pulse amplitude is too large to fall in the display range, decrease the [**Gain**] in the Calibrate tab of the *Pixie4 Run Control* Panel . Since the offsets might drift, for example after changes in input count rate, it is useful to leave the display open and check the offsets once in a while.

2. In the Calibrate tab, input an estimated preamplifier RC decay time for Tau in µs then click on [**Auto Find**] to determine the actual Tau value for the current channel of the current module. Repeat this for other channels if necessary. The Tau finder works best for a Tau value from 20 µs to 200 µs.

3. In the Settings tab, click on [**Save**] to save the system parameters found so far. You can save the settings into either an existing settings file or a new file.

4. Click on the Run tab, set [**Run type**] to 0x301 MCA Mode, [**Polling time**] to 1 second, and [**Run time/timeout**] to 30 seconds or so, then click [**Start Run**]. After the run is complete, select the Analyze tab and click on the [**MCA Spectrum**] button. The MCA spectrum shows the MCA histograms for all four channels. You can deselect other channels while working on only one channel. You can do a Gauss fit on a peak by entering values in the [**Min**] and [**Max**] fields as the limits for a Gauss fit. You can also use the mouse to drag the Cursor A and B in the MCA spectrum to the limits of the fit. Enter the true energy value in the [**Peak**] field to calibrate the energy scale.

   If you are not getting a nice-looking spectrum, you may need to adjust some settings such as filter rise time and flat top etc. Refer to the User's Manual for details.

- **Pixie-4 Control Panel**

The *Pixie4 Run Control* Panel is displayed on the desktop after starting the IGOR experiment file "Pixie4.pxp". Using the four control tabs, arranged in logical order from left to right, you can set up the system, set run parameters, take data, and view the results.

Control Tabs:

Settings
Calibrate
Run
Analyze

It also has four buttons on the bottom of the panel:

[**Start System**] can be used to re-initialize the modules
[**More**] will show additional controls on the control panel
[**Help**] will open this help file, at the entry corresponding to the currently active tab
[**About**] will show software and hardware version information

- **Settings**

The *Settings* Tab of the *Pixie-4 Run Control* Panel contains parameters that control the operation of the Pixie-4. Most settings are changed on a channel-by-channel basis. Controls in the ModuleRegisterPanel affect the module as a whole, controls in the ChassisRegisterPanel are used to set up interactions between modules. The *Settings* Tab also contains controls used to load, save, copy, and extract settings.

If not all described controls are visible, click on the [**More**] button at the bottom of the panel.

Energy and Trigger Filter

In this group of controls you can set the rise and flat top times for the [**Energy Filter**] and [**Trigger Filter**] of each channel. The units of time are µs. For a detailed description of filter parameters please refer to the User's Manual.
There are several filter [**Ranges**]; with a granularity of 0.027µs for filter range 1 (1-bit decimation), 0.053µs for filter range 2 (2-bit decimation), and so on. The trigger filter is always operated at the full ADC sampling rate. Its rise time can be varied between 27ns and 413ns. Its flat top however is valid between 0ns and 387ns. The trigger filter will most often use a flat top comparable with the average signal rise time. In applications with very short rise times a flat top of zero will give the best pileup rejection performance.

Employing a trapezoidal filter avoids the kind of ballistic deficit that occurs when a finite rise time signal is used in conjunction with a Gaussian shaper. The energy filter flat top time should thus be a little larger than the longest rise time expected. The output of the energy filter is sampled one decimated clock cycle before the end of the flat top, plus the signal arrival may jitter by up to one decimated clock cycle with respect to the decimated clock. You should therefore make the flat top two notches longer than the signal rise time.

The sum of energy filter rise time and flat top cannot exceed 127 decimated clock periods. If you type in a rise time or flat top value that violates this bound, the Pixie Viewer will adjust it accordingly.

The trigger threshold can be set in units of ADC steps. You can get an idea of what the noise in your system is by looking at the trace acquired in the Oscilloscope located on the Calibrate Tab. Use the mouse to zoom in on parts of the displayed trace(s) and estimate the noise.

Optimize Energy Filter

An optimization of energy filter settings is carried out by scanning all possible combinations of energy filter rise and flat top within their respective limits specified by the user, examining the energy resolution at each combination, and picking the optimal combination which gives the best energy resolution. Results of this optimization, including energy filter rise time, flat top and energy resolution, are stored in an output file whose name is specified by the user in the beginning of the run. Another file which has the same name as the output file but with a different extension (*.tmp) is used to store step-by-step intermediate results.

Click on [**Optimize**] to open the *Auto Optimization of Energy Filter* panel. Before starting the optimization, all conditions listed should be fulfilled first. This is to ensure that valid MCA spectra will be produced during the optimization and Gauss Fit of energy peaks on the MCA spectra will generate meaningful results. The four channels of a Pixie-4 module can be used to optimize four different Auto Scanning Limits at the same time when the same detector signal is split and input into the four channels. This would speed up the scanning of a large range of energy filter rise time and flat top. You could change the Auto Scanning Limits of each channel by changing the [**Channel**] control on the left upper corner of the *Pixie4 Run Control* panel. At any time, the Pixie-4 module used to carry out this optimization is the current module, i.e. the module indicated by the [**Module**] control on the left upper corner of the *Pixie-4 Run Control* panel. So if you want to use a different module, change the [**Module**] number before you set the Auto Scanning Limits for each channel.

Pulse Shape Analysis

The [**Trace length**] and [**Delay**] values entered in this group of controls, both in units of µs, govern the waveform acquisition. [**Trace length**] is the total length of the acquired waveform, wich can be set independent of the parameters in the Energy and Trigger Filter.You can use the delay parameter to

move the trigger point within the trace: [**Delay**] measures the trigger time with respect to the beginning of the recorded trace. For ordinary data taking the trace lengths are up to 13.6µs for each channel. The waveforms will be read in 13.3ns increments from FIFO memory.
If event dead time is a concern, the trace length should be set as short as possible. In particular, if only energies and time stamps are of interest, the trace length can be set to zero.

The [**PSA Start**] and [**PSA End**] specify the trace range for Pulse Shape Analysis (PSA). Currently the Pixie-4 supports two types of PSA: XIA_PSA and USER_PSA. XIA_PSA reports the signal arrival time by measuring the time when the trace reaches a preset percentage level ("constant fraction") of its magnitude. The preset percentage threshold can be set defined in the ChannelRegisterPanel. The arrival time is relative to the starting time of the trace. So for XIA_PSA, the PSA Start and PSA End should be set to include the rising edge of the trace. USER_PSA is a user-defined PSA value, calculated by customized user code linked to the general software.

Action Buttons

[**Channel**] (Channel Register)	ChannelRegisterPanel
[**Module**] (Module Register)	ModuleRegisterPanel
[**Chassis**] (Chassis Register)	ChassisRegisterPanel
[**Copy**] (Copy settings)	CopyPanel
[**Extract**] (Extract settings)	ExtractPanel
[**Files/Path**] (Files and path)	AllFilesPanel
[**Load**] (Load settings)	
[**Save**] (Save settings)	

ChannelRegisterPanel

= Settings -> Channel Register

Each channel has its own channel control/status register (CCSRA). Click a check box to set or clear particular bits. Setting the checkboxes will automatically modify the 16bit register shown in the Settings Tab. We give a brief description of all relevant bits here. In the *Channel Register Panel* they appear in top to bottom order.

Bit 0: [**Respond to group triggers only**]
This bit controls waveform acquisition. To stop the FIFO and store a waveform two conditions must be fulfilled. A fast trigger primes the FIFO to stop after a programmed delay, but only if by that time a valid trigger is recorded. When bit 0 is cleared the source for that valid trigger is the locally generated event trigger of this channel. When bit 0 is set, the trigger source will be a signal on the distributed DSP-trigger line. This allows for master slave operation as outlined in the User's Manual.

Triggers are always distributed within a module: Any channel with "trigger enabled" will issue a trigger for distribution, and any channel set to "group trigger" will respond to the distributed trigger, including those issued by itself. Note that a further control in the ChassisRegisterPanel determines how trigger signals are distributed between modules.

Bit 1: [**Measure individual live time**]
This bit will in almost all applications be opposite to bit 0. Its setting decides who asserts the live time control. When cleared, the DSP ensures that during the event interrupt no channel can generate another trigger and latch new event data, at least not after the coincidence time window (see HitEditPanel below). This setting is useful in master slave operation and almost in all cases where list-mode data are required. On the other hand, when channels are operating independently and only MCA information is needed (MCA mode), but not list mode data, then this bit should be set to achieve the highest throughput.

Bit 2: [**Good channel**]
Only channels flagged as good will be read out. This setting has no bearing on the channel's capability to issue a trigger. There can be a triggering channel whose data are discarded. Channels not marked as good will not be included in the automatic offset adjustment.

Bit 3: [**Read always**]
Set this bit if you want a good channel to be read out even if it did not report a hit. It cannot report valid energy or timing data in that case, but if operated in group trigger mode you will get a valid waveform. This way you can collect waveforms not biased by trigger requirements.

Bit 4: [**Enable trigger**]
You can switch on any channel's ability to contribute to the event trigger with this bit.

Bit 5: [**Trigger positive**]
For channels with triggering enabled, this causes triggering on the rising edge of the input signal when the bit is set, and triggering on the falling edge when the bit is not set. The core of the trigger/filter FPGA can only trigger on a rising edge. So, if the bit is not set the FPGA will invert the signal before storing it in the FIFO and sending it to the core.

Bit 6: [**GFLT required**]
In a larger experiment you may want to exercise control over which events to accept and which to reject
based on external logic. A logic signal inhibiting event triggers can be distributed on the PXI backplane. When bit 6 is set, a logic high on the GFLT (Veto) backplane line will inhibit latching data and issuing triggers for this channel.

Bit 7: [**Histogram energies**]
Switch on incrementing an energy histogram in the DSP's MCA memory with this bit. You can choose to have histogramming in list mode runs. The histograms will continue to be updated over multiple runs, started with the resume run command.

Bit 14: [**Estimate energy if not hit**]
If a channel is read out even if it did not report a hit (see [**Read always**]), its energy is reported as zero since there was no valid local trigger to capture the value of the energy filter. However, since the energy filter is computed continuously, setting this bit will capture the filter value based on the (last) group trigger distributed within the module or over the backplane (if sharing triggers over the backplane). This might be useful for channels with occasional very small pulses (below the threshold), or possibly to capture energy estimates on piled up pulses.
Note that since the timing of the group trigger is not precise with respect to the non-triggering pulse, the energy reported is only a rough estimate. It might help to set the flat top time to a large value to make the capturing of the energy filter less time sensitive.

Bit 10: [**Compute constant fraction time**]
The DSP can use pulse shape analysis to compute a precise signal arrival time using the digital equivalent of a constant fraction discriminator. For this to work correctly the rising part of the signal should be fully contained in the recorded trace. The time computed is the arrival time after the start of the acquired waveform in units of 1/256th of an ADC sampling interval. This information can be used to replace the recorded channel time which is derived from a (digital) leading edge discriminator, cf the User's Manual.

The DSP code shipped with the Pixie-4 has some pulse shape analysis capabilities already built in. One of these functions, the digital constant fraction discriminator takes an input parameter---the [**Threshold**] percentage. The default value is 25% since it is a commonly used threshold fraction for this type of discriminator. The result of the computation is the time of signal arrival measured with respect to the start of the acquired waveform. The result is written into the channel header in the linear output data buffer, cf the User's Manual for details.

[**Integrator**]
This variable controls the event energy reconstruction:

0: Normal code.

1: Use energy filter gap sum only. This is equivalent to simply integrating over the pulse. It is useful for scintillator applications where event energies can be derived by setting the energy filter flat top long enough to cover the whole scintillation pulses.

2: Ignore energy filter gap sum when reconstructing event energy. This is useful for step pulses whose amplitude is the difference between the high and low steps.

3-5: Same as 1, but the energy is multiplied by a factor 2,4, or 8, respectively. This is useful for very fast pulses that do not have a lot of area under the peak and thus will show up only at the very low end of the spectrum.

ModuleRegisterPanel

= Settings -> Module Register

The Module Register Edit Panel controls several module-wide parameters.

Edit Required Hit Patterns

In this section, you can select the *Coincidence Pattern* that defines which channels have to contribute to an event to make it acceptable. For example, you might be only interested in events in which only a single channel was hit, or in any event in which more than two channels contributed. This feature is useful for Pixie-4 channels operating independently from each other, though they may be sharing clocks and triggers.

For each event, the Pixie-4 first reads which channels contributed before reading any further data. This *Hit Pattern* is a 4 bit number, e.g. (0001) if only channel 0 contributed or (1111) if all channels contributed. By setting the checkboxes you can require that an event must match one of the selected hit patterns to be accepted for further processing. Any or all hit patterns can be selected; if none is selected no events will be accepted, if all are selected any event will be selected. The hit pattern (0000) can be useful when sharing triggers between modules: traces etc will be recorded in the module for channels with the "Read Always" Bit set in the ChannelRegisterPanel even if there is no hit in the module itself. Setting the checkboxes will automatically modify the 16bit coincidence pattern shown in the Settings Tab, where it can also be edited directly.

An example shall illustrate this feature. Assume a single module connected to 4 detectors which observe a Na-22 source, emitting back to back 511keV gamma-rays from positron annihilation. Channels 0 and 1 are connected to one pair of back to back detectors and channels 2 and 3 are connected to a second pair of back to back detectors. You are interested only in gammas from positron annihilation. Thus a coincidence in channel 0 and 1 or a coincidence in channel 2 and 3 is required. If all 4 channels were in coincidence, that would be fine too. So, the acceptable hit patterns would be (0,0,1,1), (1,1,0,0) and (1,1,1,1), where the right most digit indicates channel 0 and the left most is for channel 3. To achieve the desired behavior, you have to select the three acceptable hit patterns in the HitEditPanel by checking the appropriate boxes, and deselect all other hit patterns by not checking their boxes.

Note the difference between "hit" and "trigger": A "hit" occurs if a pulse goes over threshold and passes pileup inspection. "Hits" can not be disabled. A "trigger" means there is a "hit" plus a request for event processing is sent to the module's DSP unit. "Triggers" can be disabled on a channel by channel basis in the ChannelRegisterPanel. Channels will be recorded if they have a hit or the [**Read always**] bit is set in the ChannelRegisterPanel, but only "hit" channels will have valid energy and timing information.

Coincidence Window

Each channel with a pulse above threshold, whether trigger enabled of not, contributes to the hit pattern the moment the pulse is validated as not piled up (i.e. energy filter time after rising edge of pulse). The hit pattern is read for comparison with the coincidence pattern about 160 ns after the *first* pulse is validated. If several channels contribute to an event, the minimum coincidence window -- the time period in which (delayed) channels can contribute to the hit pattern -- is thus ~160ns. When sharing triggers over the backplane, the minimum width is ~400ns.

A difference in peaking times between channels will cause even channels with simultaneous pulses to contribute to he hit pattern at different times. The software thus calculates the required additional window width to compensate for any such difference, displays it in the [**Required additional width**] field, and ensures that the [**User added width**] is at least this value. If longer delays between channels are expected from the physics of the experiment, this added width can be increased up to a value of ~200ms.

If the required additional with is later decreased by reducing the difference in peaking times, a smaller coincidence window is possible. However, to avoid modifying a large coincidence window intentionally set by the user, the [**User added width**] is never *decreased* automatically.

Notes:
1) Any added coincidence window width will increase the time required to process an event and thus reduce the maximum count rate.
2) In run types 0x100-0x300, pulses contributing *during the readout of data* (after the end of the coincidence window) are lost. The readout may take several microseconds, longer if waveforms are to be recorded.
3) The cut off at the end of the coincidence window is precise to within 13.3ns * 2^(Filter Range), e.g. ~100ns in range 3.

Group Trigger Timestamp

Set this checkbox to record individual timestamps for all channels in group trigger mode, rather than the identical time stamps based on the (last) group trigger.

Normally, in acquisitions with shared group triggers, all channels record the (identical) timestamp of the (last) group trigger for this event. Since waveforms are captured based on the group trigger, this ensures that within a data record traces and timestamps are correlated. However, if no waveforms are recorded, there is no time-of-arrival information of possible delays between channels. Setting the checkbox preserves the time difference information. See the user manual for details.

Clover Addback

To support 4-fold clover detectors, there is a further option to sum energies from individual channels in events with more than one hit and bin them into an addback spectrum. This function is enabled with the checkbox [**Sum channel energies for addback MCA**]. The Pixie-4 module will thus generate 4 channel MCA and one addback MCA at the same time.
If clover addback is selected, there is the further option to bin into the individual channel MCAs either energies from all events, or only energies from events with hits in a single channel.

Clover addback is currently only supported in MCA mode.

ChassisRegisterPanel

= Settings -> Chassis Register

The Chassis Setup Panel controls several system-wide parameters.

Backplane Options

If [**Group trigger**] operation is requested in the ChannelRegisterPanel, each channel will respond to distributed triggers rather than its own local trigger. Triggers are always distributed within a module: Any channel with [**Trigger enabled**] will issue a trigger for distribution, and any channel set to [**Group trigger**] will respond to the distributed trigger, including those issued by itself.
In addition, triggers can be distributed over the PXI backplane between modules. If the checkbox [**Share triggers with other modules**] is checked, the module issues and responds to triggers from other modules in the system. If not, the module is disconnected from the backplane; group triggers are only distributed between the 4 channels of the module. If no channel is set to respond to group triggers, this checkbox should be cleared. Rev. B modules are always connected to the backplane.

One option to connect an external GFLT (Veto) signal to the Pixie-4 system is to use the MMCX front panel connector on a Pixie-4 module. The GFLT signal is distributed via the PXI backplane to all modules, and - if enabled in the ChannelRegisterPanel - a channel only records events where the GFLT signal is logic 0 (0V). If the [**Use front panel for GFLT input**] checkbox is set, the current module will internally connect to front panel input labeled "DSP-OUT" to drive the GFLT backplane line. Only one device may issue the signal to the backplane. If the checkbox is set for one module, the function will automatically be disabled for all other modules. However, it is the user's responsibility to ensure that no other device in the PXI chassis issues signals on this line.
The GFLT signal must be a LVTTL signal, i.e. 0 = 0V, 1 = 3.3V.

A second option for the MMCX front panel input is to configure it to contribute to the STATUS line on the backplane. This is a wire-OR line connected to all modules in a backplane segment. If the MMCX input is high, the module will pull down the wire-OR line (this constitutes a logic 1 for this line). The state of the STATUS line can be recorded during event processing (see below)

If coincidence information is to be shared between neighboring modules, a module can combine the result of its own coincidence test with the result received from the left (or right) neighbor (form an AND) and pass it on to the right (or left) neighbor. In this way, the leftmost module in a chain can know if *all* modules to the right passed their coincidence test. This function is enabled by setting the [**Always add input from left/right neighbor to output to right/left neighbor ...**] checkboxes. Even if a module is not enabled to distribute its own coincidence test, it can still transmit test results from its neighbors.

Module Coincidence Setup

Each module always tests the *Hit Pattern* of contributing channels against the user defined *Coincidence Pattern* (set in the ModuleRegisterPanel) before recording and processing an event. In addition, the module can be set up to share the results of its coincidence test with other modules in the system and accept events only if also the pattern of modules passing or failing their coincidence test is acceptable according to a user defined *Module Pattern*.

This module coincidence function is enabled by setting the checkbox [**Perform module coincidence test**]. If it is not checked, the controls underneath are colored in the background color to indicate that their status has no effect.

The [**Share Pattern**] defines those *Hit Patterns* for which the module signals a pass or fail of its coincidence test to the other modules. The definition of the *Share Pattern* is equivalent to the *Coincidence Pattern* and can be found using the FindPatternPanel. Usually, the [**Share Pattern**] will be identical to the [**Coincidence Pattern**] (which is shown for reference only, it can be edited in the

ModuleRegisterPanel), so that a module simply signals to its neighbors if it passed the local coincidence test. However, in some applications it might be useful to signal and locally accept for different hit patterns.

The next five checkboxes define where the module shoud send the result of the *Share Pattern* test:
- to the left neighbor
- to the right neighbor
- to the STATUS line
- to the TOKEN line
- to slot 2 using a dedicated Star Trigger line

STATUS and TOKEN are wired or lines, they are logic 1 if at least one module signals a "pass" on this line. The STATUS line can in addition be set to logic 1 by an external signal via the MMCX front panel input. The line to slot 2 is useful if a device in slot 2 assembles a "hit pattern of module" from all slots and sends a derived value such as "more than N modules hit" to the STATUS or TOKEN line.

The [**Module Pattern**] defines which combination of module coincidence lines are acceptable for recoding of events. The definition of the *Module Pattern* is similar to the *Coincidence Pattern* and can be found using the FindPatternPanel.

In summary, the procedure for module coincidences is as follows: When an event trigger is issued to the DSP in a module, the DSP will first check if the local Hit Pattern is acceptable according to the local Coincidence Pattern (test 1). Then the DSP checks if the Hit Pattern is acceptable according to the Share Pattern (test 2) and signals the result of test 2 to the other modules. After a short delay, the DSP reads the results of test 2 in the other modules from the backplane lines, and tests if the line status is acceptable according to the Module Pattern (test 3). If both test 1 and test 3 are passed, the event is recorded, else data acquisition resumes.

FindPatternPanel

= Settings -> Chassis Register -> Find

In this panel, you can determine a) the *Share Pattern* that defines which channels have to contribute to an event to signal a pass to the other modules and b) the *Module Pattern* which backplane lines have to be logic 1 to make an event acceptable.

a) For each event, the Pixie-4 reads which channels contributed in a module. This *Hit Pattern* is a 4 bit number, e.g. (0001) if only channel 0 contributed or (1111) if all channels contributed. If module coincidences are enabled and events with e.g. hit pattern (0011) = 3 are to be shared, bit 3 in the *Share Pattern* must be set. When setting or clearing the checkbox corresponding to one or more hit patterns in the panel, the *Share Pattern* is displayed at the bottom.
The pattern found must be manually entered in the share pattern fields of the ChassisRegisterPanel.

b) If module coincidences are enabled, the Pixie-4 reads the status of 4 backplane lines, which form a 4 bit number similar to the channel hit pattern. The difference is that channels (3, 2, 1, 0) are replaced by lines (TOKEN, STATUS, Right, Left). If events with e.g. backplane status (0011) = 3 are to be recorded, bit 3 in the *Module Pattern* must be set. When setting or clearing the checkbox corresponding to one or more hit patterns in the panel, the *Module Pattern* is displayed at the bottom.
The pattern found must be manually entered in the share pattern fields of the ChassisRegisterPanel.

CopyPanel

= Settings -> Copy

This panel can be used to copy parameter settings from one module to another. The source module and channel are selected at the top of the panel. The parameters to be copied are organized into list box in the left-hand column. The right-hand column shows the destination channels and modules for the copy operation. The Items to copy shown on the Copy Panel and the actual variables to be copied

are listed below.

Items	Actual variables to be copied
[Gain]	Gain [V/V]
[Offset]	Offset [V] and base percent
[Filter]	Energy Filter Rise Time and Flat Top, Baseline Cut
[Trigger]	Trigger Filter Rise Time and Flat Top, Trigger Threshold
[FIFO]	Trace Length, Delay, dT [µs], PSA Start, PSA End, CFD Threshold
[ChanCSR]	Channel CSRA and Channel CSRB
[Coinc.]	Coincidence Pattern, Coincidence Window
[MCA]	Cut-Off Energy, Binning Factor
[TAU]	Tau [µs]
[Intergrator]	Integrator
[ModCSR]	All module coincidence settings and backplane options except front panel GFLT

After selecting source, destination and parameters, click on the [Copy] button to execute the copy operation.

ExtractPanel

= Settings -> Extract

This panel can be used to extract parameter settings from a file to selected modules and channels. The source file is specified at the top of the panel. Click on the [Find] button to locate the source file. Parameters to be extracted and destination modules or channels are selected in the same manner as in the CopyPanel. Click the [Extract] button to execute the operation.

AllFilesPanel

= Settings -> Files/Paths

This panel gives you access to the underlying files of the Pixie-4 software. Usually, these files are already loaded in the memory of the Pixie-4 Viewer. You only have to change these files when you receive updates from XIA.

The directory locations are specified as complete (not relative) search paths: the DSP Path for the DSP code; and the FPGA Path for the trigger/filter FPGA configuration. Use a colon (:) as the separator between drive name, directory, and subdirectories. Do not use backslashes (\). For example use "D:XIA:data" rather than "D:\XIA\data".

File and Path names should not exceed 80 characters.

- **Calibrate**

If not all described controls are visible, click on the [More] button at the bottom of the panel.

Analog Signal Conditioning

In the Analog Signal Conditioning section you can set the gain and DC offset for the selected channel and module. Note that the ADCs are dc-coupled to the Pixie-4 inputs, and thus compensation for any DC-offset is necessary. You will rarely have to set this manually, as the DC-offsets can be adjusted automatically through clicking on [Adjust Offsets] on the Oscilloscope.

The gain and offset settings are given in units of V/V and V, respectively. The voltage gain computed is the ratio between the pulse height at the module input to the pulse height at the ADC input. Note that

the ADC has a 2.2V input range.

Histogram Control

This section shows the parameters controlling the operation of the multichannel analyzer built into the DSP memory. Energy values are calculated to 16-bit fixed-point numbers. This would correspond to a 64k spectrum.

To map the full energy range into the available 32k spectrum, one has to combine bins. At minimum, 2 bins have to be combined into one, so the [**Binning Factor**] has to be set to 1 (combining 2^1 bins). Higher binning factors can be useful for low count rate or low resolution applications.

If you want to see a certain range of the spectrum at higher resolution you can enter a [**Minimum Energy**]. This will discard all energies below the minimum and start binning the spectrum from the minimum energy (Bin 0 = Emin). The minimum energy is not applied in MCA runs.

Decay Time

The [**Decay Time**] is the exponential RC time constant of the preamplifier. It is required in order to properly calculate corrections to measured energy values. To set and measure the decay time, enter an estimated value then click on the [**Auto Find**] button. You can also enter a known good value directly in the control. The RC calibration needs to be performed only once for a given preamplifier. The result is then stored in the parameter database, and can be saved in the settings file by clicking on the [**Save**] button in the Settings tab.

Manual Fit

Manual Tau Fitting is done on a channel-by-channel basis. Clicking [**Manual Fit**] opens the *TauDisplay*. First, choose the channel on which you want to do the manual fit by changing the [**Channel**] control on the left upper corner of the *Pixie4 Run Control* panel. Then set proper [**dT**] on the *TauDisplay* panel, and click the [**Run**] button to read an untriggered trace for the chosen channel. The exponential fit range can be set either by putting the Cursors A and B on the trace or changing the [**Fitting_start**] and [**Fitting_end**] controls on the *TauDisplay* panel. Clicking the [**Do exponential fit**] button will perform the exponential fit on the portion of the trace specified by the fitting range. If possible, some of the baseline after the pulse should be included in the fit region. The fitted tau is going to be reported in the [**Fitted tau**] control and the deviation between the fitting curve and original ADC trace is shown in the [**Deviation**] control. Click [**Tau OK**] to download the Tau value to the Pixie-4 module.

Optimize Tau

Clicking [**Optimize**] opens a panel that allows automatic scanning of all Tau values within the scanning limit specified by the user, examining the energy resolution at each Tau, and picking the optimal Tau value which gives the best energy resolution. Results of this optimization, including the Tau value and energy resolution, are stored in an output file whose name is specified by the user in the beginning of the run. Another file which has the same name as the output file but with a different extension (*.tmp) is used to store step-by-step intermediate results.

Before starting the optimization, all conditions listed on the *Auto Optimization of Decay Time* panel should be fulfilled first. This is to ensure that valid MCA spectra will be produced during the optimization and Gauss Fit of energy peaks on the MCA spectra will generate meaningful results. The four channels of a Pixie-4 module can be used to optimize four different [**Decay Time Limits**] at the same time when the same detector signal is split and input into the four channels. This would speed up the scanning of a large range of Tau values. You could change the [**Decay Time Limits**] of each channel by changing the [**Channel**] control on the left upper corner of the Pixie-4 Run Control panel. At any time, the Pixie-4 module used to carry out this optimization is the current module, i.e. the

module indicated by the [**Module**] control on the left upper corner of the Pixie-4 Run Control panel. So if you want to use a different module, change the [**Module**] number before you set the [**Decay Time Limits**] for each channel.

- **Oscilloscope**

  = Calibrate -> Oscilloscope

  The Oscilloscope shows 8192 untriggered ADC samples from the input for each channel. The time between samples can be set using the [**dT**] variable. The display is updated through its [**Refresh**] button. The DC offset of the preamplifier signal has to be compensated for in order to bring the DC-coupled input into the ADC range. The exact DC value has no bearing on the acquired spectrum and its origin, which is always at zero. The DC-adjustment is used only to ensure that the signals to be measured fall comfortably into the ADC range. When clicking the [**Adjust Offsets**] button, the Pixie-4 Viewer will set the DC offset to a percentage of the full ADC range specified in the [**Baseline**] control.

  The offset calibration must be performed with the preamplifiers connected to the Pixie-4 inputs and with both the preamplifier power and detector HV switched on. One should also repeat the offset calibration each time measurement conditions change in any major way, e.g., when the count rate changes greatly. All such changes may influence the DC offset value of the preamplifier signal.

  To analyze the noise spectrum of the acquired trace, click on the [**FFT Display**] button, which opens the FFTDisplay.

  To view the effect of the digital filters applied to the ADC traces, click on the [**Show Filters**] button which opens the ADCFilterDisplay .

- **FFTdisplay**

  You can analyze the noise spectrum in the trace captured in the Oscilloscope, by observing the Fourier transform of the signal. For best results, remove any source from the detector and only regard traces without actual events. The chart shows a plot of amplitude vs. frequency. The plot is calibrated such that a sine wave with 100 ADC units amplitude (200 units peak-to-peak) will show up with an amplitude of 100. To convert a noise floor measurement into ADC units/sqrt(Hz) use the variable FFTbin displayed at the top of the chart, which tells the width of each frequency bin in the Fourier spectrum. The conversion from amplitudes to rms ADC units/sqrt(Hz) is accomplished by multiplying with 1/sqrt(2*FFTbin). Now, observe that an ADC unit corresponds to 61μV. Using the known gain of the Pixie-4 you can convert the noise into an input noise voltage density measured in V/sqrt(Hz). Or, given a particular energy calibration, the noise density can be expressed as eV/sqrt(Hz).

  If you click on the [**Apply Filter**] button, you can see the effect of the energy filter simulated on the noise spectrum.

- **ADCFilterDisplay**

  This graph shows the ADC trace for the selected channel and the response of the (slow) energy filter and the (fast) trigger filter together with an estimate of the trigger threshold. The display is updated through its [**Refresh**] button.

  Notes:
  1. For best representation of the filters, the filter lengths should be an integer multiple of the ADC sampling interval.
  2. Only the simple trapezoidal difference filter is shown for the energy filter. In the actual pulse height

calculation, corrections are applied that take into account the decay of previous pulses, the contribution during the flat top time, and long terms baseline effects. These corrections are not applied to the energy filter shown in the ADCFilterDisplay.

- **Run**

If not all described controls are visible, click on the [**More**] button at the bottom of the panel.

Run Type

This popup menu is used to set the run type to one of the following modes:

List Mode

List mode is the general data acquisition run. Waveforms, energies and time stamps are collected on an event-by-event basis. The data is stored in various formats (see section 3.6 of the user manual for details):

0x100	full event data (9 words), plus waveforms
0x101	full event data (9 words), no waveforms
0x102	compressed event data (4 words), no waveforms
0x103	compressed event data (2 words), no waveforms

Since available memory limits the number of events that each module can store in its buffer, the Pixie-4 Viewer computes the maximum number of events. When the maximum is reached, the run is stopped and the buffer is read out. For a longer run in list-mode, you can request several spills, or buffer fills. For example, if you request a run with 10 spills, you will get 10 list mode buffers worth of data. At start of the first run all previous run history is cleared. For instance MCA memory and run and live time information are cleared. The next nine sub-runs are started with a Resume Run command, which leaves previous run information intact. Run times and live times and spectra in MCA memory are incremented.

You can also manually adjust the maximum number of events stored before the run is stopped. Some data acquisition systems, which are geared towards event-by-event readout and are not able to handle large buffers, may benefit from the capability to reduce the maximum number of events per spill.

Fast List Mode

Fast list mode is an event-by-event data acquisition run without waveforms. Since no traces are read out, the data acquisition is faster than a regular list mode; however, no pulse shape analysis (PSA) values are available. There will also be no run statistics, no coincidence testing, and no checks if the event buffer is filled faster than the event processing rate. So the average trigger rate must be kept well below the processing rate. Otherwise, the data from the remainder of the run will be corrupted. The data is stored in various formats (see section 3.6 of the user manual for details):

0x200	full event data (9 words), no waveforms
0x201	full event data (9 words), no waveforms
0x202	compressed event data (4 words), no waveforms
0x203	compressed event data (2 words), no waveforms

Fast List Mode is mainly a legacy run type, and only marginally faster (slightly shorter dead time from event processing) than a normal list mode run recording waveforms of zero length.

MCA Mode

MCA mode puts all modules into a typical spectrum-only acquisition mode in which there are no list-mode data required. The event data is not stored in the output buffer, but only used to calculate the

energy for incrementing the spectrum. Runs end after the time specified in the "RunTime/TimeOut" control counts down to zero. The “Maximum no. of Events” control is set to zero for MCA runs since it is not used to end the run.
Any [**Minimum Energy**] entered in the Histogram Control is ignored for MCA runs.

Polling Time

The polling time indicates the time interval at which the Pixie-4 Viewer checks if the run in the selected modules has ended. If so, runs are stopped in all modules, if they have not stopped already, and the data are read out.

Run Time/Time Out

This variable is used to indicate the total run time for MCA runs or the timeout limit for list mode runs. During a run, it counts down the remaining time.

Number of Spills

The variable indicates the number of repeated runs. It is only used in list mode runs. Even if set to zero, there will be at least one spill.

Maximum no. of Events

This variable indicates the maximum number of list-mode events the Pixie-4 module can store in its buffer for each run. It should be zero (indicating unlimited events) for MCA runs.The number is calculated automatically if you change the run type, but it can be reduced manually.

Synchronization

The first check box asks if all runs should [**Simultaneously start and stop**] in all modules. In almost all multi-module systems this will be the case and the box should be checked.
Synchronization signals are distributed over a PXI backplane line. If a Pixie-4 module is present in the crate, but not part of the current data acquisition setup, it might inhibit the synchronization setup.

If you also want all timers in all modules to be reset to zero with the start of the next data acquisition run, click the box [**Synchronize clocks**]. For this feature to be useful all Pixie-4 modules should be operating from the same master clock as described in the user's manual.
Normally, once clocks are synchronized, they will stay synchronized until modules are power cycled. Therefore the checkbox is cleared at the end of the run, preserving time correlation between subsequent runs. In some cases it may be beneficial to resynchronize the clocks in every new run (i.e. reset the clock to zero for every new data file). To do so, check the [**in every run**] checkbox.

Output File

You can choose a [**Base name**] and a [**Run number**] in order to form an output file name . The run data will be written to files whose name is composed of both (i.e. base####.xxx). The run number is automatically incremented at the end of each run if you select [**Auto increment run number**] on the Data Record Options panel, but you can set it manually as well. Data are stored in files in either the MCA folder if the run is a MCA run or the PulseShape folder if the run is a List Mode run. These files have the same name as the output file name but different extension as described below.

".bin"
For list mode runs, buffer data are stored in a file with name extension ".bin". This is a binary file consisting of 16bit unsigned integers.

".dat"
For list mode runs, a summary of event data is stored in a file with name extension ".dat". This is an ASCII file. For long list mode runs with many spills, this file can become much larger than the binary file. If you do not need he ASCII data, you can disable the ".dat" file by unchecking the [**Auto process list mode data ...**] in the Data Record Options panel.

".mca"
For both list mode runs and MCA runs, MCA spectrum data are stored in a file with name extension ".mca" if you select [**Auto store spectrum ...**] on the Data Record Options panel. This is a binary file consisting of 32bit unsigned integers. It includes MCA data for all modules, ordered from channel 0 of module 0 to channel 3 of the last module

".set"
Module settings are stored in a file with name extension ".set" after each run if you select [**Auto store settings ...**] on the Data Record Options panel. Run statistics can also be extracted from this file. This is a binary file consisting of 16bit unsigned integers. It is equivalent to the settings files saved from the Settings tab.

".ifm"
For both list mode runs and MCA runs, run statistics information are stored in a file with the extension ".ifm" if you select [**Auto store statistics ...**] in the Data Record Options panel.

Start Run

After setting all parameters, you can start a run to take data. During the run, the Run Time/Time Out control shows the remaining time for MCA runs or time out count down for list mode runs. If you select multiple spills for list mode runs, the number of spills will also count down during the run.

For list mode runs, when the first module reaching the preset maximum number of events stops its run, it will also stop the runs in all other modules if Module Synchronization is enabled. Then the data buffer of each Pixie-4 module will be read out and saved into a file. If more than one spill is requested, the run will resume in all modules.

For MCA runs, when the Run Time/Time Out control counts down to 0, the Pixie-4 Viewer will issue a run stop command to stop the runs in all modules. Then the MCA histogram of each module will be read out and saved into a file.

Stop Run

If you want to stop a run before it finishes by itself, you can click on this button to manually stop it. This will end runs in all modules and read out and save the data.

Data Record Options

This panel gives you several options for automating tasks after or during a run. They are all checked by default to ensure all data are saved for each data run.

[**Auto increment run number**] will increase the run number in the file name of the data files to avoid overwriting of files.
In addition, if this option is checked, Igor will test at the beginning of a run if the output file already exists (testing the .bin file for list mode runs and the .mca file for MCA runs). If the file exists, Igor will increment the run number and try again; after 20 tries it will add "_new" to the base file name and continue. This feature is intended to avoid overwriting previous files when Igor only remembers a run number from an earlier run, for example after a PC crash.

[**Auto store spectrum ...**] will store the spectrum data automatically after a run.

[**Auto store settings ...**] will store the run parameters automatically after a run in binary format (same as settings file). Run statistics can also be extracted from this file.

[**Auto process list mode data ...**] will extract the energies and timestamps from the binary data and save it in an ".dat" file in ACSII format.

[**Auto store statistics ...**] will store the run statistics and run start/stop date and time automatically after a run in ASCII format in an .ifm file.

In addition, you can choose to automatically [**Update MCA every N seconds**] during MCA runs or every N spills during List mode runs. "N" can be selected in the variable control field. Run statistics will be updated at the same time. You can also manually update the MCA spectrum by clicking the [**Update**] button in the MCA Spectrum display.

Further, you can choose to automatically store data in [**New Files every N spills**] during List mode runs or every N seconds during MCA mode runs. "N" can be selected in the variable control field. Run statistics will be updated at the same time. This feature can be used to break up output data into several smaller files; since it will take some time to save the data, N should be set to a rather large value, i.e. at least several hundred spills or several thousand seconds.
Note that a *new* run will be started after saving, i.e. all run statistics will be reset, MCA spectra will be cleared, and clocks resynchronized if the the [**in every run**] checkbox is set in the Synchronization section in the Run Tab. Thus enabling this feature is equivalent to manually starting several individual runs with N spills.

A further option changes the data readout for list mode runs. If checked, the module stores 32 data buffers in external memory, which then are read out in a fast block read. Each readout counts as one "spill". The total number of events will be 32 times the Maximum no. of events times the Number of spills shown in the *Run* Tab. External memory readout is not supported for Rev. B modules.
If not checked, buffers are read one by one from internal memory. The total number of events will be the Maximum no. of events times the Number of spills shown in the *Run* Tab. The readout deadtime will be higher in this mode.

[**Do not parse list mode file ...**] will disable the automatic parsing of the list mode file after the run to build a list of pointers to individual events. This is required for Igor to display events in the List Mode Traces graph, but may take a long time for large files. If disabled, you have to manually load the file in the List Mode Traces graph before cycling through events.

- **Analyze**

On the top left part of the Analyze tab shows the run time and the measured event rate for the selected module. The right part shows for each channel the live time and the input count rate. Note that the run time is the sum of time spent in sub-runs (called spills), but ignoring the time it took the host to read out the data from Pixie-4 modules. Similarly, the live time was measured only while one of the sub-runs was ongoing.

Besides the [**Update**] button to refresh the run statistics (only), there are four buttons in the middle of the tab opening the following graphs and panels:

MCA Spectrum

Pulse-height spectra accumulated in the internal Pixie-4 memory can be displayed after pressing the MCA Spectrum button. Pulse heights are computed to 16 bits precision, i.e. correspond to 64k spectra. As the memory allows for only 32k words per channel, bins have to be combined according to the Binning factor for each channel in the Histogram Control.

You can select the module you want to inspect and you can add or remove individual channel displays by clicking the MCA check boxes. The [**Sum MCA**] checkbox controls display of the clover addback spectrum. Gray fields indicate values can be edited.

The [**Fit**] menu allows you to make Gaussian fits to peaks in the histograms. The fit range can be set channel by channel in the [**Min**] and [**Max**] fields, or by placing cursors on the spectrum with the mouse. The [**Fit**] menu starts a fitting routing for one or all channels. The routine does take a constant background term into account, though its value is not displayed. The fit results that are displayed include the peak position, the number of counts in the peak, and its relative and absolute full width at half maximum (FWHM), calculated from the Gaussian fit. For best results be sure to extend the fit range to cover some of the constant background.

To calibrate your energy scale, you can after the fit type the true energy value into the field [**Peak**] and the scale will automatically be adjusted.

The [**Sum**] menu gives you three options of summing the spectrum. You can either sum the range defined by the [**Min**] and [**Max**] fields without any corrections, sum the range and subtract the background, or sum the entire MCA. Summing is performed on all four channels at the same time. The calculated Sum is displayed in the [**Peak Area**] field, and the MCA maximum of the sum range is displayed in the [**Peak**] field.

The [**Files**] popup menu allow to store individual spectra and read back stored spectra from disk. There are three operations:
[**Save MCA to Igor text file**] will save the current MCA (4 channels) as a scaled wave with some added comments in a text file. This can be useful to import data into other applications
[**Read MCA to Igor text file**] will read back a Igor text file saved as above.
[**Extract MCA from binary file**] will read MCA data for the current module from a binary file, such as the
.mca file automatically saved at the end of the run. The file is assumed to be in 32bit unsigned integer
format.

You can [**Update**] the MCA during a run at any time. The [**MCA source**] field displays where the MCA was last read from, e.g. from the module's memory or a specific file.

The [**Zoom**] buttons above the graph can be used as shortcuts to zooming operations with the mouse. The [**Reset Scale]** button resets the scaling of the MCA to 1/bin, i.e. it undos any calibration entered through the [**Peak**] field.

List Mode Traces

After a list mode run has finished, the acquired waveforms and event data can be displayed on an event-by-event basis in the List Mode Traces panel. The most recently acquired data file will be searched for the event requested in the [**Trace number**] field. The display will show the traces from the selected module, and the associated time stamps, energies and PSA values for those channels that reported a hit in this event. Traces and energies are scaled as 16-bit numbers. Note that the ADC traces shown in the Oscilloscope are raw 14-bit numbers, i.e. ADC traces have values divided by 4.

In order to display traces from an earlier experimental run one needs to change the Data File name by entering it directly in the [**Data File**] control or clicking the [**Find**] button.

To see the response of the energy and trigger filters for the current event, click on [**Digital Filter**], which opens the Filter Display. This window is mainly intended for diagnostic purposes. For information how to use the Pixie-4 for more detailed pulse shape analysis, please contact XIA.

Filter Display

This graph is used to see the simulated response of the energy and trigger filters in an event. You can browse the leading edge trigger filter response and the energy filter response of individual events. The latter requires a trace length of at least twice the peaking time plus the gap time to be displayed. The trace is shown in red. The trigger filter is shown in blue, and the energy filter is shown in green.

Note that only a simple difference filter is displayed for the energy filter, without the corrections for baseline and decay of the pulse used in the energy calculations.

List Mode Spectrum

Pulse height spectra can be reconstructed from list mode data stored on the disk. The file shown in the [**Data File**] field will be processed and the resulting histograms will be displayed for the selected Pixie-4 module. Use [**Read**] after changing the data file to process the new data, and [**Histo**] to update the displayed spectrum. The full spectrum length is equal to 64k channels. Use [**No. of bins**] and [**Delta E**] settings to compress the spectrum such that it fits the display. Hint: use 8000 and 4 to see the full range of data, and then adjust these numbers to zoom into the range of interest. The number of bins and the delta E variables are kept in memory for each channel individually. Be sure to select the channel of interest prior to changing these variables. Use the mouse to zoom in on peaks of interest.

The [**Fit**] menu allows you to make Gaussian fits to peaks in the histograms. The fit range can be set channel by channel in the [**Min**] and [**Max**] fields, or by placing cursors on the spectrum with the mouse. The [**Fit**] menu starts a fitting routing for one or all channels. The routine does take a constant background term into account, though its value is not displayed. The fit results that are displayed include the peak position, the number of counts in the peak, and its relative and absolute full width at half maximum (FWHM), calculated from the Gaussian fit. For best results be sure to extend the fit range to cover some of the constant background.

The [**Zoom**] buttons above the graph can be used as shortcuts to zooming operations with the mouse. The [**Reset Scale]** button resets the scaling of the MCA to 1/bin, i.e. it undos any calibration entered through the [**Peak**] field.

All Run Statistics

The run statistics for all modules and channels are displayed in the *All Run Statistics* panel. It also shows the time and date of run start and stop and the source where the statistics were last read from. The information can be refreshed during a run by clicking the [**Update**] button, it can be saved to or read from an .ifm file with the [**Files**] menu.

The "DAQ Fraction" is an estimate of the fraction of the lab time a module is taking data. As described above, "Run Time" and "Live Time" measure only the time a module was actively taking data, but not the time to setup a run or the time a module is waiting for readout by the host. Thus "Run Time" is always smaller than the elapsed "Lab time" (i.e. the difference between date/time of run start and run stop). "DAQ Fraction" is defined as "Run Time" / (Current date/time - run start date/time) *100 and thus gives an estimate of the time lost for readout etc. (After a run is finished, the run stop time replaces the current time)

Note that "Run Time" is read from the module, while the date/time is obtained from the host computer to a precision of 1s. "DAQ Fraction" is therefore not a precise number.

In list mode runs, the time to read out one spill is about 0.025s per module. The DAQ Fraction will drop significantly if the time to fill the memory approaches this value. To maximize the DAQ Fraction, a) run in 32 buffer/spill mode, b) make sure the polling time is significantly smaller than the fill time, c) define only those channels actually used as "good", and d) minimize the amount of waveform captured or run in a compressed list mode to store more events in the same amount of memory.
For example, with 4 channels active in mode 0x103, the memory will hold 744*32 = 23,808 events, so at a count rate of 23kcps, the memory will fill in about 1s and be read out in less than 0.1s, which results in a DAQ Fraction of more than 91%